Computer Networks 159 (2019) 96–109

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Adversarial environment reinforcement learning algorithm for intrusion detection

Guillermo Caminero, Manuel Lopez-Martin*, Belen Carro

Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011, Spain

ARTICLE INFO

Article history:
Received 31 October 2018
Revised 29 March 2019
Accepted 16 May 2019
Available online 17 May 2019

Index terms:
Intrusion detection
Reinforcement learning
Adversarial learning

ABSTRACT

Intrusion detection is a crucial service in today's data networks, and the search for new fast and robust algorithms that are capable of detecting and classifying dangerous traffic is essential to deal with changing threats and increasing detection difficulty. In this work, we present a new intrusion detection algorithm with an excellent prediction performance. The prediction is based on a classifier which is a simple and extremely fast neural network. The classifier implements a policy function that is trained with a novel reinforcement learning model, where the behavior of the environment is adjusted in parallel with the learning process.

Intrusion detection frameworks are based on a supervised learning paradigm that uses a training dataset composed of network features and associated intrusion labels. In this work, we integrate this paradigm with a reinforcement learning algorithm that is normally based on interaction with a live environment (not a pre-recorded dataset). To perform the integration, the live environment is replaced by a simulated one.

The principle of this approach is to provide the simulated environment with an intelligent behavior by, first, generating new samples by randomly extracting them from the training dataset, generating rewards that depend on the goodness of the classifier's predictions, and, second, by further adjusting this initial behavior with an adversarial objective in which the environment will actively try to increase the difficulty of the prediction made by the classifier. In this way, the simulated environment acts as a second agent in an adversarial configuration against the original agent (the classifier). We prove that this architecture increases the final performance of the classifier.

This work presents the first application of adversarial reinforcement learning for intrusion detection, and provides a novel technique that incorporates the environment's behavior into the learning process of a modified reinforcement learning algorithm.

We prove that the proposed algorithm is adequate for a supervised learning problem based on a labeled dataset. We validate its performance by comparing it with other well-known machine learning models for two datasets. The proposed model outperforms the other models in the weighted Accuracy (>0.8) and F1 (>0.79) metrics, and especially excels in the results for the under-represented labels.

1. Introduction

Considering the importance of current security attacks on modern highly demanding networks, the economic importance of services running on these networks and the increased demands on data networks imposed by these services, it is very important to rely on automatic systems capable of detecting intrusions in a fast and reliable manner. Such a system is an Intrusion Detection Systems (IDS) [1], being its final goal to fast and accurately analyze network traffic and to predict potential threats.

IDS is identified as one of the main applications of machine learning research to new data networks [2–6]. IDS presents important prediction problems, since it must handle large, noisy, and unbalanced datasets. Moreover, the features extracted from the network traffic are complex and usually with a noisy assignment of labels to its corresponding ground-truth state, due to the difficulty to ascertain the true value of the intrusion state. This difficulty is the reason why the association between the features and the intrusion labels is done manually.

The data used to train an IDS algorithm are normally provided as a dataset of recorded network features with their associated intrusion labels. The network features are usually recorded in an automatic manner with a post-manual assignment of their associated labels. This dataset of network features and their associated

* Corresponding author.
E-mail addresses: gcaminerof@gmail.com (G. Caminero), mlopezm@ieee.org (M. Lopez-Martin), belcar@tel.uva.es (B. Carro).

https://doi.org/10.1016/j.comnet.2019.05.013
1389-1286/

intrusion labels is the basic element to train all classic Machine Learning (ML) models in a supervised learning framework. However, this is not the framework used to train a Deep Reinforcement Learning (DRL) algorithm [7], which is based on the interaction of an agent with an environment. This environment receives actions from the agent and returns new states and rewards [7], which are used by the algorithm to optimize the agent's policy function. The policy function is used by the agent to compute the next best action, based upon the current state and reward (Markov property) [8] provided by the environment. The ultimate goal of the DRL framework is for the agent to produce actions that maximize the total sum of rewards granted by the environment, along a trajectory of interactions between the agent and the environment. To translate the DRL terminology to an IDS problem, we can assimilate: 1) the states to network traffic samples, 2) the actions to label predictions and 3) the rewards to a value associated with the goodness of the prediction, i.e. classification accuracy (better prediction implies a greater reward value). Considering the differences between supervised and DRL frameworks, the reinforcement learning (RL) model needs an evident adaptation, because we want to use it with a dataset of pre-recorded samples formed by network features and associated intrusion labels.

Never used before in IDS, a Reinforcement Learning (RL) algorithm can provide very interesting properties for intrusion detection: (1) It is a very general framework with a flexible reward function, that does not require to be differentiable. (2) Once the training is done, the resulting policy function is usually a simple and fast neural network. (3) Thanks to the use of simple reward functions, the algorithm is suitable for online training, which allows a rapid response to changes in network conditions.

Considering the above points, we propose a new model that integrates a simulated environment, which provides samples of network traffic and rewards with an agent, which implements the classifier, and which is trying to predict the correct intrusion label based on the network samples given by the environment. The rewards generated by the environment will be positive/negative depending on the correct/incorrect prediction of the agent. The algorithm is trained with the objective of maximizing the total sum of rewards. An important characteristic of this simulated environment is that it randomly extracts the new samples (states) from the dataset of pre-recorded samples. This initial strategy of random extractions provides good results, however we have designed for the environment a more sophisticated behavior that is learned during training, acting against the agent's policy (adversarial), so better results may be obtained. That is, the environment behavior is actively trying to reduce the rewards given to the agent, by increasing the classifier's incorrect predictions and forcing it to learn the most difficult cases. The resulting new algorithm is then called Adversarial Environment using Reinforcement Learning (AE-RL). How the sampling process and reward function is implemented by the modified environment constitute the specific nature of the proposed model. AE-RL is particularly suited to incorporate a supervised problem which make use of a labeled dataset into a DRL framework.

One of the problems of RL algorithms (when dealing with labeled datasets) occurs when the dataset is not properly balanced. This fact is important in all machine learning algorithms, with a special emphasis on the RL algorithms, since they do not know a priori the correct label used to adjust the weights and minimize the error function. These algorithms learn from transitions through trial and error, therefore when a dataset is highly unbalanced, it is difficult for the algorithm to classify under-represented samples without incurring in over-fitting for the rest of the data. Facing this dilemma as the main motivation, we propose AE-RL. With this objective, the method aims to maximize the total sum of rewards by obtaining a balanced learning for classification tasks. For this purpose, the intelligent environment is used to provide the most interesting samples based on an exploration-exploitation principle that is present in all reinforcement learning algorithms. The resulting sampling strategy permits a more balanced learning for the less frequent samples. This novel approach allows us to overcome the exploration problems in these algorithms and to deal with unbalanced datasets.

To evaluate the performance of AE-RL, we have compared it with a series of well-known and widely used supervised machine learning models: Linear and Radial Basis Function (RBF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Gradient Boosting Machine (GBM), Random Forest, AdaBoost, Multinomial Logistic Regression and Convolutional Neural Networks (CNN), and several classic DRL algorithms: Double Deep Q-Network (DDQN), Dueling DDQN and Asynchronous Advantage Actor Critic (A3C). These algorithms cover most of the different areas of machine learning that are used to implement classifiers: neural networks, linear models, kernel methods, bagging and boosting models, deep learning and reinforcement learning models. In particular, it is important to highlight the differences between the performance results of AE-RL with the other DRL algorithms, which have a related but not similar nature. A detailed presentation of results is provided in Section 4.

The AE-RL algorithm produces a *dynamic* sampling process which usually implies an over-sampling/under-sampling of the minority/majority classes of an unbalanced dataset. This is achieved with the intelligent sampling provided by the simulated environment and is quite different from other classic *static* over-sampling/under-sampling algorithms. Classic over-sampling methods create new samples close in "distance" to existing samples that belong to some specific minority class (e.g. Synthetic Minority Over-sampling Technique, SMOTE) [9], and additionally can give a higher "weight" to samples that are harder to learn (closer to a majority class) (e.g. Adaptive Synthetic Sampling, ADASYN) [10]. Similarly, the under-sampling techniques reduce the number of samples of the majority classes, either by random under-sampling or by substituting a number of samples by a representative one (either original or synthetic). Considering these analogies, it is interesting to compare the performance results of AE-RL with these other classic over-sampling/under-sampling methods. This comparison is provided in Section 4. The *static* over-sampling/under-sampling methods produce the new samples prior to training. In contrast, the model proposed for AE-RL generates the samples *dynamically* during training and depends on the training process. Additionally, the samples produced by many over-sampling/under-sampling methods are synthetic, that is, new samples that are related to the original ones but not identical. In contrast, AE-RL does not generate synthetic samples, so it better preserves the original information contained in the data set.

It is also interesting to appreciate the differences of AE-RL with the bagging (e.g. Random Forest) and boosting (e.g. AdaBoost, GBM) algorithms, when dealing with the problems of generalization and dataset imbalance [11]. AE-RL modifies the training frequency of each sample based on its training error, dealing with imbalance and providing excellent generalization errors (Section 4: accuracy and F1 metrics on the test dataset). Boosting methods are also directed by the sample training error, but they operate by changing the weights given to each sample, increasing the weights of the samples with larger errors in a series of additive models (ensemble model). Boosting deals well with generalization and imbalance. Bagging is based on training a series of models using random sampling with replacement (bootstrap) of the original dataset, the final model is based on averaging the results of all the models (ensemble model). In the case of bagging, the successive resampling of the dataset is based on random sampling and it is not based on training results. Bagging deals well with generaliza-

tion but not directly with imbalance unless additional cost factors are introduced.

To carry out the experiments, we have chosen two well-known datasets: NSL-KDD [12] and AWID [13]. These intrusion detection datasets have been extensively studied, contain a sufficient number of samples to obtain significant results and are highly unbalanced, with most of the samples associated with a few labels. Both datasets are divided into training and test sets. All performance results for both datasets are provided over the test sets.

As a summary, the motivations/contributions of this work are:

- To propose a new classifier model for intrusion detection in networking.
- To present a classifier which is fast, flexible and with excellent performance metrics for prediction.
- To integrate the RL framework with a supervised classification problem.
- To explore the important field of research of multi-agent and adversarial RL, and its application to intrusion detection.
- To apply, for the first time, an adversarial RL to address the training bias associated with an unbalanced dataset. And, to provide a comparison of the results of our proposed model with other models which deal with this imbalance problem using static (e.g. SMOTE, ADASYN) methods.

The paper is organized as follows: Section 2 identifies related works. Section 3 presents the dataset used for the experiments and the model proposed in detail. Section 4 shows the results obtained and finally, Section 5 provides discussion and conclusions.

2. Related works

There are many works in the literature presenting results for intrusion detection using different datasets. Nevertheless, comparing results is a difficult task, since an important source of differences relies on the test set used to obtain the detection scores. In many published works it is different from the original test set, or it is not clear which one was used.

As far as we know, there are no previous works applying DRL for intrusion detection with the same premises used in this work, however many consider existing machine learning and deep learning models. In this section we present the most representative of these works applied to the NSL-KDD and AWID datasets.

We also discuss the most significant works applying reinforcement learning (RL) to intrusion detection and classification in general, and related works that, applying techniques other than DRL, adjust the class distribution of a dataset by over-sampling/under-sampling to achieve a balanced training.

Machine learning and intrusion detection: Intrusion detection can be addressed as anomaly detection [14] or as a classification problem, depending on the available data. In this analysis we focus on the classification perspective. Many works apply ML for intrusion detection with the NSL-KDD dataset: in [15] they report an accuracy of 79.9% for test data for the 5-labels prediction scenario, applying an MLP with three layers. Authors in [16] report an F1 of 98% for the 5-labels scenario using AdaBoost with Naive Bayes as weak learner and a previous feature selection; test results are based on 10-fold cross validation over the training data, not on the test set. Finally, in [17] a variational autoencoder is proposed to perform detection on NSL-KDD with a 5-labels configuration, obtaining an overall accuracy of 80%.

Considering the AWID dataset, the original work that first presented the dataset [13] is an excellent source of results for different ML models for intrusion detection. In this work, the authors apply 8 algorithms (AdaBoost, Hyperpipes, J48, Naïve Bayes, OneR, Random Forest, Random Tree and ZeroR) to the dataset. They perform a thorough comparison of the models, and the J48 model presents the best results reporting an accuracy of 96% and a F1 of 94.8%.

Recently, a survey in [4] presents a complete review of the ML models for intrusion detection in IoT networks.

Reinforcement learning and intrusion detection: The work in [18] presents an anomaly detector based on reinforcement learning with a simulated network environment, where anomalies are injected in a controlled manner and the reward is based on the correct detection of the anomalies. From a logical point of view, this experiment presents similarities with the present work: although the environment is physically simulated, the reward function is manually controlled and is not generated by the environment itself, and the real-time generation of sequences of actions, states and rewards could be assimilated to those registered in a dataset. However, the techniques applied in [18] notably differ from the present work, since they use a Q-learning algorithm based on a look-up table, that requires a discretization of states to avoid an explosion in the table size. In contrast, current DRL models use function approximators based on neural networks (NN), which allow generalizing their application to states of any size with continuous or discrete values. In [19–21] they also employ look-up tables with temporal difference (TD) learning [22] for intrusion detection in live sequences of traffic flows.

Multi-agent reinforcement learning and intrusion detection: Authors of [23,24], in a different framework perform intrusion detection by applying reinforcement learning based on a multi-agent architecture, forming a hierarchy to detect anomalies. The configuration of agents is not adversarial. They require the discretization of the state space to apply a look-up table. None of the above works apply DRL models to intrusion detection. The results provided apply to its own simulated network and cannot be compared with the results of this work.

Reinforcement learning and classification: In [25] a DRL approach based on an actor-critic model to perform classification over several well-known UCI datasets (e.g. Iris, Hepatitis…) is presented. Considering the applied technique this work is similar to ours; however, there are differences in the taken approach to perform classification, based on extending the state space dimension with additional ancillary variables (features) used as memory cells, in a copy and erase process together with a complex reward scheme. This has a major impact on complexity and computational performance (e.g. training times) and follows a very different strategy from the one presented in this work, which applies simple reward functions and does not require feature engineering. Also, in [26] a demonstration of classification by reinforcement learning is shown, and it is compared with traditional classifiers such as SVMs obtaining similar ratios without the need of feature engineering. Authors in [27] present an actor-critic reinforcement learning model based on Temporal Difference (TD) learning, applied to classification using a fuzzy adaptative learning control network. The resulting model is applied to the very simple Iris dataset.

Adversarial reinforcement learning: There are excellent overviews of multi-agent and adversarial reinforcement learning, and none of them present results for intrusion detection [28–30]. Current applications are mainly found in the field of games or related commercial application (e.g. e-negotiation systems).

In [31] an adversarial environment is described, in which the objective is a classifier that makes mistakes by performing small modifications to the training data. This framework is not applicable to the type of classification provided in this paper, whose intention is to make correct classifications instead of forcing an error in the algorithm.

Adversarial in cyber environments: In [32] a simulated network environment as a cybersecurity zero-sum game with incomplete information is presented, where two adversarial agents

(attacker and defender) try to win the game. The result of the game cannot be transformed into a classifier and the applied methods for the policy of reinforcement learning are Monte Carlo and Q-learning (based on a look-up table), so not DRL. Authors in [33] introduce an adversarial environment based on reinforcement learning where a defender and attacker dynamically adjust their behavior. The work focuses on cybersecurity attacks and is specifically aimed at the Heartbleed attack. The study is mainly theoretical and is based on an ad-hoc RL algorithm; the results are based on numerical simulations not on real data.

Over-sampling of unbalanced datasets for intrusion detection: In [34] a complete analysis of several over-sampling algorithms applied to the NSL-KDD data set is provided. [34] proposes a new over-sampling method called Variational Generative Model (VGM) based on a variational autoencoder. This work applies the new method together with 7 variants of SMOTE and ADASYN to the NSL-KDD dataset (5-labels scenario) to generate synthetic data that is used to train several well-known classifiers (Random Forest, Logistic Regression, Linear SVM and MLP). The classifiers obtain the best performance increase when generating new data with VGM, producing a significant increase in performance compared with no-oversampling. Even considering this new over-sampling technique, the results obtained with AR-RL outperform the performance metrics (Accuracy and F1) obtained with VGM for all classifiers. Section 4 provides a complete comparison between the results obtained in [34] and the results of AE-RL.

3. Work description

In this Section, (1) we describe the datasets chosen to compare the detection capacity of the different models, and (2) provide a comprehensive review of our proposed model (AE-RL).

The datasets employed are described in Section 3.1. The proposed model is presented in detail in Section 3.2.

3.1. Selected datasets

To verify the capabilities of the proposed model for intrusion detection under different conditions, we have chosen two well-known datasets in the field of intrusion detection: NSL-KDD [12] and AWID [13].

Each one offers the opportunity to evaluate the performance of the model under a different number of intrusions and attacks distribution.

3.1.1. NSL-KDD dataset

The NSL-KDD [12] dataset is a classic well-known IDS dataset. NSL-KDD dataset is an evolution of the KDD-99, solving the redundant records problems of the original dataset. The NSL-KDD dataset has 125973 training samples and 22544 test samples, containing 41 features: 38 continuous and 3 categorical (discrete valued). These features have been additionally transformed: scaling all continuous features to the range [0–1] and one-hot encoding all categorical features. This provides a final dataset with 122 features: 38 continuous and 84 with binary values ({0, 1}) associated to the three one-hot encoded categorical features. This is a very unbalanced dataset with a frequency of 43.1% and 1.7% for the most and least frequent labels.

Each training sample has a label output from 23 possible labels (normal plus 22 labels associated to different types of anomaly). The test data has the same number of features (41) and output labels from 38 possible values. That implies that the test data has anomalies not presented at training time. The 23 training and 38 testing labels have 21 labels in common; 2 labels only appear in training and 17 labels are unique to the test dataset. Up to 16,6% of the samples in the test dataset correspond to labels unique to the test dataset, and which were not present at training time. The existence of new labels at testing introduces an additional challenge to the learning methods.

To facilitate interpretation of results the 23 labels have been aggregated into meaningful categories. As presented in [12], the training/testing labels can be associated to one of five possible categories: NORMAL, PROBE, R2L, U2R and DoS. All the above categories correspond to an intrusion except the category: NORMAL, which implies that no intrusion is present. We have considered these five categories as the final labels driving our results (Section 4). These new labels are useful for characterizing intrusions, maintaining a fairly unbalanced distribution (an important feature of intrusion data) and with a number of samples, in each category, large enough to provide significant results.

To visualize the unbalanced nature of the NSL-KDD dataset, Fig. 1 shows the frequency distribution of the intrusion classes (on the left) and the types of attacks (on the right) for the NSL-KDD

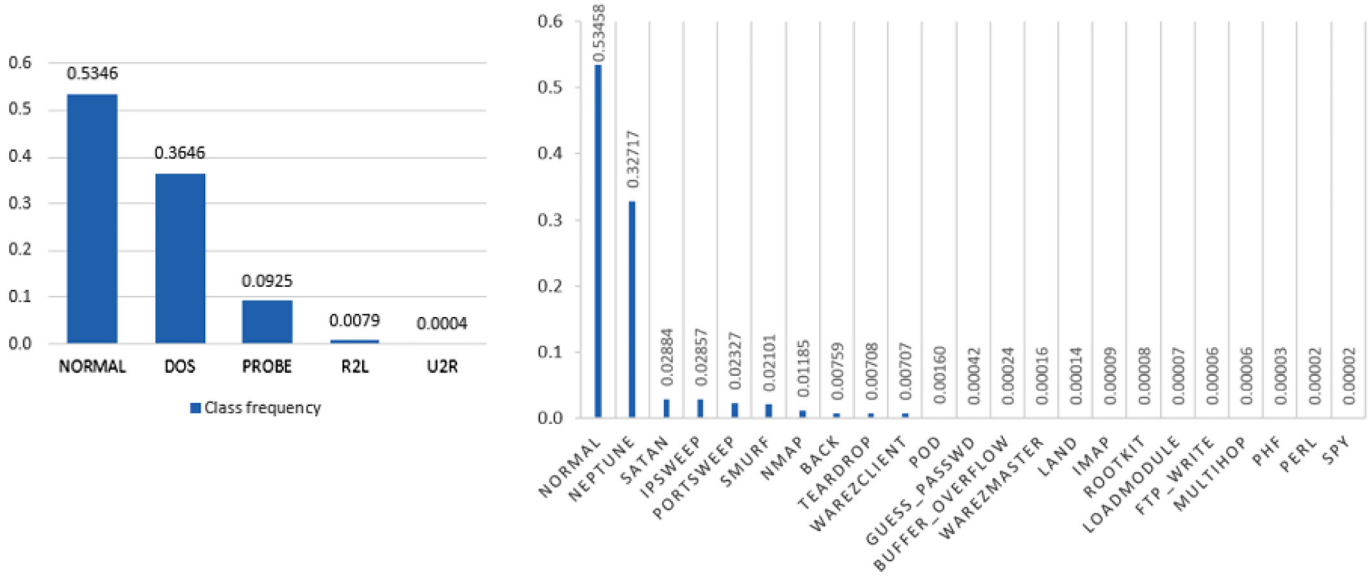


Fig. 1. Frequency distribution of intrusion classes (left) and types of attacks (right) for the NSL-KDD dataset.

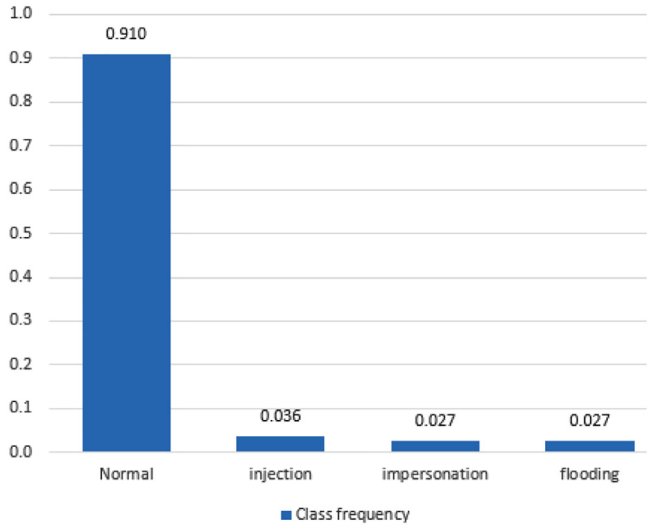


Fig. 2. Frequency distribution of intrusion classes for the AWID dataset.

training dataset. In the analysis performed here, we have 5 possible intrusion classes and 23 possible attack types

3.1.2. AWID dataset

The publicly available Aegean WiFi Intrusion Dataset (AWID) [13] contains a rich blend of normal and attack traffic against IEEE 802.11 networks. AWID is larger and more recent than NSL-KDD.

AWID offers two different groups of data. We have chosen the AWID-CLS-R that includes a separated training and test datasets. The dataset provides a class label with 4 values: normal, flooding, injection and impersonation; the first corresponds to a non-attack situation and the rest to different types of attacks. The dataset contains 154 features (continuous and categorical) with 1.795,574 and 575,642 samples for the training and test datasets, respectively. After discarding the features with null values, constant values and network addresses we reduced their number to 24 features (continuous and categorical). The continuous features have been scaled to the [0–1] range and all categorical features are one-hot encoded.

This is also a very unbalanced dataset with 91% of normal samples and 9% associated with anomalies: 2.7% flooding, 2.7% impersonation and 3.6% injection. Fig. 2 clearly shows the imbalance in the class distribution of the AWID dataset.

3.2. Model description

We propose a novel model called AE-RL that implements a classifier based on the theory of RL [8]. In an RL framework, an environment informs an agent of the state of the environment and the agent responds with an action within the environment. This action eventually produces a change in the state of the environment. The environment responds to the action of the agent with a reward that is associated with how good/bad the action is in terms of some final goal. Considering this framework of an agent that interacts with a live environment, our proposed model provides the following modifications:

- We provide a simulated environment whose states correspond to random samples taken from a dataset of labeled attacks (network intrusions).
- The agent implements a classifier which tries to predict the intrusion label from the states provided by the simulated environment.
- The simulated environment produces rewards according to the correct/incorrect predictions of the agent.

The new framework obtained by the previous modifications allows applying a well-known RL algorithm (Q-learning algorithm) [8] to classify intrusions using a dataset of pre-recorded intrusion data.

The Q-learning algorithm is based on finding the best Q-function for the agent (the classifier in our case). A Q-function estimates a value for each state-action pair, this value corresponds with the sum of the rewards for the given state considering that we take a certain action and then move forward with the current policy. An important result is that the Q-function can be calculated iteratively by the following expression [8]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_A Q(S_{t+1}, A_{t+1}) - Q(S_t.A_t)\right] \quad (1)$$

Where S_t is the current state, A_t is the current action, A_{t+1} is the next action, R_{t+1} is the next reward value, α is the learning rate and γ is the discount factor, which in this case is set close to zero because the states are not correlated with each other (they are obtained by sampling the dataset, not in a sequential order), and therefore there is no need for the algorithm to remember previous states. Values of 1.0 and 0.001 for α and γ have been used, respectively.

In order to approximate the Q-function, a fully connected neural network (NN) is used, following [35]. The input to this NN is the current state, which corresponds to the features extracted from the labeled dataset. The output of the NN represents the Q-function for the set of available actions. As mentioned earlier, the Q-function corresponds to how good is taking some action in the current state. In [35] the Deep Q-Network (DQN) algorithm followed by the AE-RL algorithm (see Table 1) is described. The DQN model does not apply the expression in (1) to update the Q function, but a related one based on a quadratic loss function: $(R_{t+1} + \gamma \max_A Q(S_{t+1}, A_{t+1}) - Q(S_t.A_t))^2$ (Table 1).

The framework that results from the above premises, based on the Deep Q-Network algorithm, provides good detection results (Section 4: Fig. 7; results for the DDQN model). However, considering the unbalanced nature of the dataset, where normal traffic is highly over-represented, we have changed this original framework to provide the environment with an intelligent behavior beyond the random sampling of the dataset. This change allows us to address the problem of the unbalanced dataset with a new model that performs a dynamic and intelligent resampling of the dataset during training.

In this new model (AE-RL), we associate a new reinforcement learning agent to the environment. Both agents, the environment agent and the classifier (main) agent, are trained in parallel (both using DQN). The action (output) of the classifier agent will be the intrusion type prediction for a sample of the dataset (input), and the action of the environment agent will be the class of attack that will be used to generate new samples in the training process. Therefore, even when both agents have a similar structure, the main agent acts as a classifier and the environment agent acts as a selector of the attacks that will be used in the next round of training. The two agents work in an adversarial model based of the rewards offered to the (main) agent which are inverted for the environment agent. In this way, the environment agent will try to direct the generation of samples to increase the errors produced by the classifier agent (reduce its rewards), forcing the classifier to focus on the most difficult samples.

With this framework, the new model generates a more balanced data sampling by producing samples in which the classifier fails more frequently. This may be due to the low frequency of occurrence in the training set or to the difficulty of prediction for some states. All positive rewards for the main agent will

Table 1
Algorithm for the new model AE-RL.

Algorithm for AE-RL:

Initialize $Q_c(s, a_{c_t})$ arbitrarily
Initialize $Q_e(s, a_{e_t})$ arbitrarily
Repeat (for each episode):
  Initialize the state value: s_0 = random sampling (dataset)
  Choose environment's initial action a_{e_t} using policy derived from $Q_e(s_t, a_{e_t})$
  Replace s_t using s_t = random sampling ($S(a_{e_t})$), where $S(a_{e_t})$ are all samples whose label is a_{e_t}
  Repeat (for each time step, t→ 0 to N):
    Choose agent's action a_{c_t} using policy derived from $Q_c(s_t, a_{c_t})$
    Take RL step obtaining (r_{c_t}, r_{e_t},s_{t+1}):
      Obtain rewards for the agent and the environment: r_{c_t}, r_{e_t}
      Obtain next state:
        Choose next environment's action $a_{e_{t+1}}$ using policy derived from $Q_e(s_t, a_{e_t})$
        Replace s_{t+1} using: s_{t+1} = random sampling ($S(a_{e_{t+1}})$), where $S(a_{e_{t+1}})$ are all samples whose label is $a_{e_{t+1}}$
    Q function update:
      Apply gradient descent on the loss function given by: $(r_{e_t} + \gamma \max_{a_{e_{t+1}}} Q_e(s_{t+1}, a_{e_{t+1}}) - Q_e(s_t, a_{e_t}))^2$
      Apply gradient descent on the loss function given by: $(r_{c_t} + \gamma \max_{a_{c_{t+1}}} Q_c(s_{t+1}, a_{c_{t+1}}) - Q_c(s_t, a_{c_t}))^2$

be negative for the environment. In this way, the environment learns which are the classes in which the main agent fails most frequently and with a certain probability increases the frequency of these samples. To do this, we use two different Q-functions: a $Q_c(s, a)$ function responsible of the optimization of the classifier and another $Q_e(s, a)$ function for optimizing the environment. Both functions refer to the same principle of how good it is to take an action at a certain state, but the number of actions considered can be different. In this case, for the NSL-KDD dataset, the Q-function for optimizing the classifier has a set of actions ($A_c \in [0-4]$) corresponding to the number of classes of the classifier. On the other hand, the Q-function responsible for optimizing the environment has a set of actions ($A_{e_{train}} \in [0-22]$ *or* $A_{e_{test}} \in [0-37]$) corresponding to each of the possible attacks in the training/test data set.

The policy followed throughout the training corresponds to a decreasing epsilon-greedy so that the exploration is initially high, reducing its value along the episodes. We consider an episode as a training round throughout the entire dataset, in this case another usual denomination for episode is epoch. Both the agent and the environment choose their actions considering their policy, being the lower bound of epsilon different for each case. In order to maximize detection, the lower bound of epsilon for the classifier policy will be low, while the lower bound of the environment policy will be set as a hyperparameter.

The method followed throughout the training corresponds to the following sequence (Table 1): (1) The Q functions of the environment and the classifier agents are initialized to a random value. In addition, an initial s_0 state is randomly chosen among all the states in the dataset in order to feed the Q function and obtain the action values for this state. It is important to keep in mind that a state is a sample from the dataset. (2) Next, the environment chooses an action a_{e_t} (intrusion label) based on its policy and the current state. (3) The environment selects the current state randomly from the dataset s_t, whose action corresponds to the one chosen by the environment, shown as $S(a_{et})$ in Table 1, which provides the feature-label pair (s_t, a_{e_t}) (4) Given the state selected by the environment, the agent tries to classify this state based on its own policy and assigns it to an action (a_{c_t}) being this step the same as a normal DQN algorithm. (5) This action: a_{ct} (intrusion label) is sent to the environment and compared with the ground-truth label, if both are the same, the correct classification provides a positive reward to the agent and if they are different the positive reward will be for the environment. (6) The new state is given by the environment as in a typical DQN algorithm, based again on its action value function and following its policy, providing the next feature-label pair $(s_{t+1}, a_{e_{t+1}})$. (7) With the reward values obtained and the next states inferred, the policy functions of both the classifier and the environment agents are updated according to the DQN update rule [35].

The reward function applied has been a simple 1/0 reward, with a value of +1 for a positive reward and 0 for a negative reward. In addition to the 1/0 reward, other reward functions were tested, in particular, the functions cross-entropy and categorical hinge were used to evaluate the distance between the predicted and ground-truth actions (labels), associating that distance with a reward (shorter distance equals greater reward). Another reward function that was examined was to provide a different reward according to the type of attack. The simplest 1/0 reward was finally selected due to better performance.

Table 1 shows with a gray background the steps of the algorithm that deviate from a normal DQN algorithm. These additional steps make it possible to train the environment agent (with an adversarial strategy).

In the definitive algorithm implemented by AE-RL we added three additional refinements to the one presented for simplicity in Table 1: a) The final algorithm implemented by AE-RL is based on DDQN [36], which is a variant of DQN using two separate networks to select and to evaluate an action. b) The actual loss function used to train the agents is based on a Huber loss, which is similar to a quadratic loss up to a threshold and linear beyond that value. The objective of the Huber loss is to reduce the values of the gradients that could experience explosive behavior. c) Since we have two agents implemented by two different neural networks, we apply two epsilon-greedy strategies. We start both networks with a high epsilon value that is reduced during training

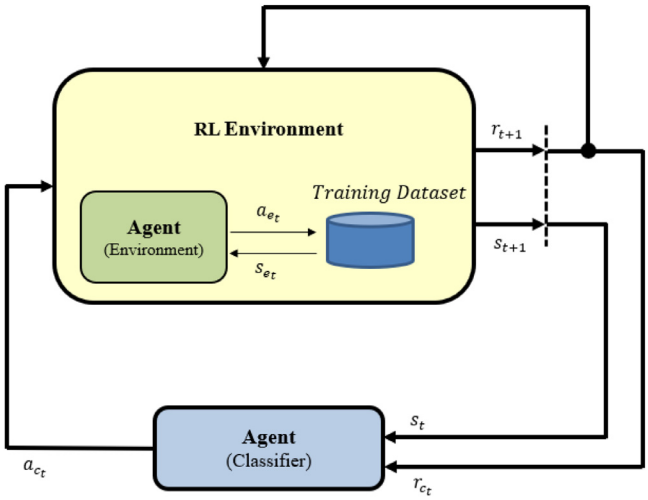


Fig. 3. Reinforcement learning interaction between the agent and its intelligent environment.

to a final value of 0.8 for the environment network and 0.01 for the classifier network. We have empirically learned that having a high value of epsilon for the environment (active exploration) is important to achieve good results.

Fig. 3 shows the elements that integrate the algorithm and how the environment is acting as a pseudo-agent that works in an adversarial mode in relation to the main agent (lower part of the diagram) that constitutes the classifier.

Fig. 3 presents a high-level view of the AE-RL algorithm, more detail is provided in Fig. 4 that graphically depicts the functional blocks and steps enumerated in Table 1. Fig. 4 separates the operation of the algorithm in the training and prediction stages. The upper part of Fig. 4 represents the training stage, where the blocks with the slightly yellow background are associated to the operation of the environment. The diagram shows the ongoing operation, and s_0 represents the initial conditions where the initial state is taken randomly from the dataset. The prediction stage (lower part) is exclusively based on the already trained classifier agent. Both the environment and classifier agents are based on very simple neural networks, as shown in Fig. 5.

Fig. 5 shows graphically the architecture on which both the attacking agent (environment) and the defending agent (classifier) are based. In both cases, the architecture is composed of a simple shallow neural network with 1 or 3 layers and 100 units per layer. The simplicity of this architecture provides a competing response time while the reinforcement learning training optimally adjusts weights and biases. In this figure it is possible to appreciate that both the attacker and defender have the features of the current sample as their input, but producing different outputs depending on the network. For the defender, the output is a prediction of the intrusion class (one of 5). For the attacker, the output is one of the 23 possible attacks that will be used to choose a random sample that corresponds to this output attack (one of 23). Therefore, the neural network for the defending agent implements a classifier, but the neural network for the attacking agent cannot be seen as a classifier but as a generator of attack classes.

Fig. 6 shows the evolution of the distribution (histograms) of attacks generated by the attacking agent during training throughout different epochs (training iterations), for the NSL-KDD dataset. The histograms only show the 23 possible attacks for the training dataset. Initially (0 epoch), the environment provides attacks randomly. As the training continues, the environment agent learns and sends attacks that maximize its reward, which varies over time. We can observe that the frequency of the different classes of attacks changes in a stochastic way, but it shows a tendency to increase the importance of several attacks: *satan, ipsweep* and *warezclient*, reducing at the same time the importance of normal traffic. This is exactly the behavior we want to observe in a dynamic (intelligent) algorithm that tries to compensate for the bias training produced by an unbalanced data set.

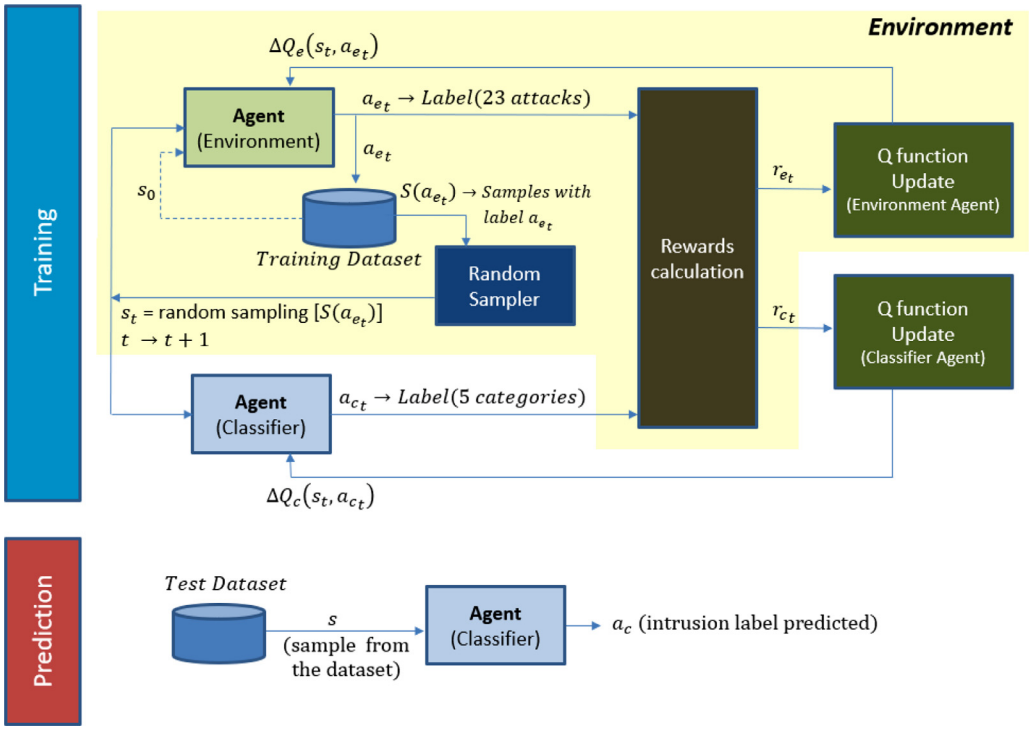


Fig. 4. Details of the AE-RL algorithm for the training and prediction phases.

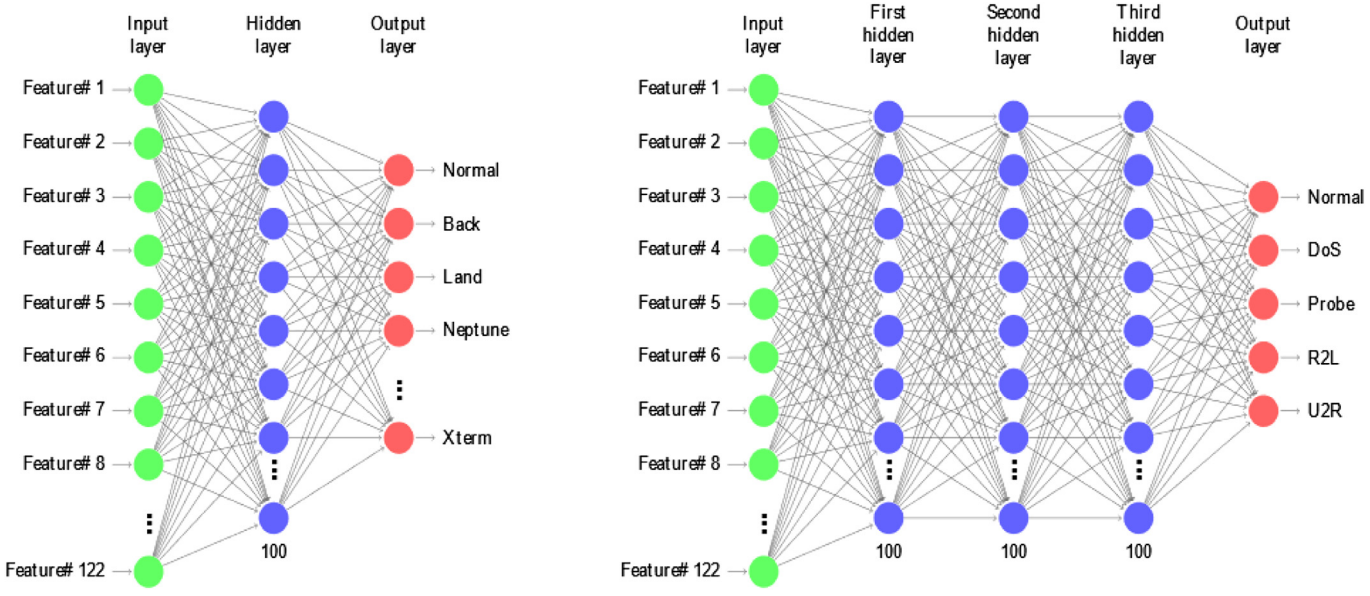


Fig. 5. Neural network architecture for the attacking agent (environment) on the left and the defending agent (classifier) on the right.

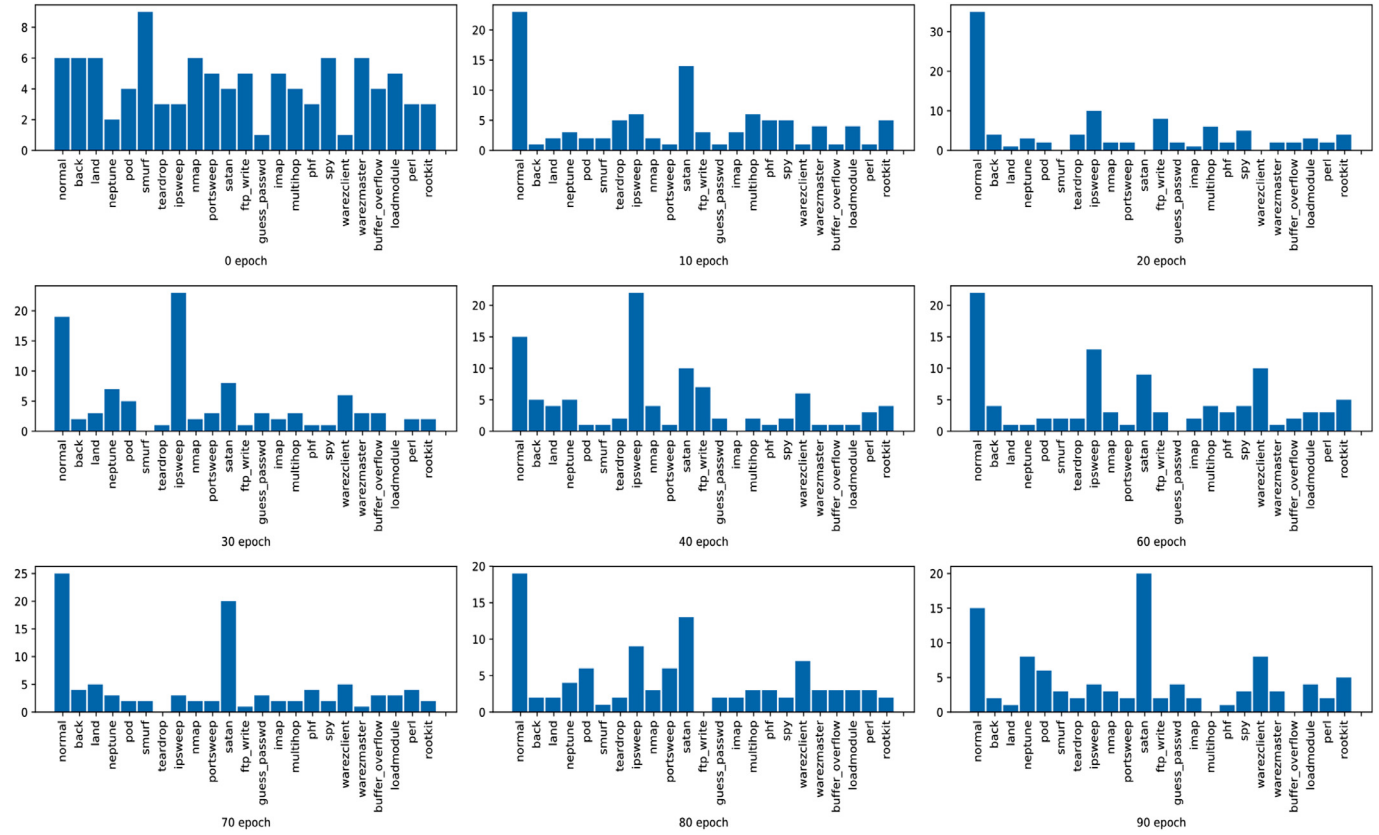


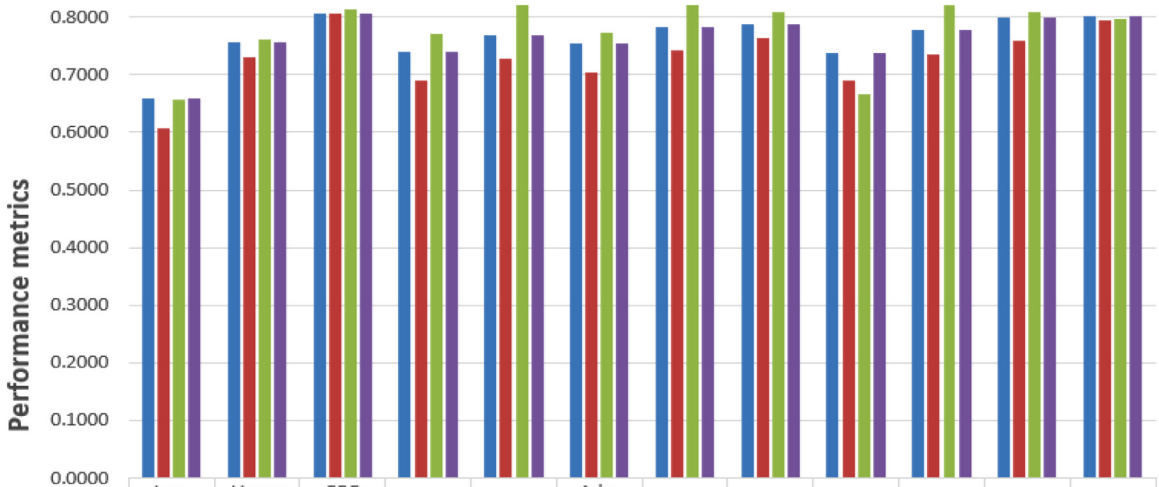
Fig. 6. Evolution of the distribution of attacks generated by the attacking agent (environment) during training (NSL-KDD dataset).

We have chosen the NSL-KDD dataset for Fig. 6 because it provides more types of attacks than AWID, which makes it easier to appreciate the redistribution of attack frequencies that the environment performs.

It is interesting to compare the frequency of attack types (right part of Fig. 1) that are present in the training dataset with the distribution of attacks generated by the environment agent of AE-RL (bottom-right of Fig. 6). This comparison provides a clear view of how the intelligent environment actively modifies the unbalanced distribution of samples to improve the classification results.

4. Results

In this section, we compare the results of applying our proposed algorithm (AE-RL) with different machine learning models to the chosen IDS datasets: NSL-KDD and AWID. We have selected some of the most common machine learning, deep learning and



	Log. Regre.	SVM Linear Kernel	SVM RBF Kernel	RF	GBM	Ada-Boost	MLP	CNN-1D	DRL DDQN	DRL Dueling	DRL A3C	AE-RL
Accuracy	0.6602	0.7560	0.8065	0.7391	0.7676	0.7531	0.7831	0.7875	0.7372	0.7788	0.8000	0.8016
F1	0.6066	0.7295	0.8056	0.6909	0.7284	0.7044	0.7429	0.7633	0.6898	0.7358	0.7600	0.7940
Precision	0.6570	0.7622	0.8130	0.7708	0.8252	0.7728	0.8262	0.8094	0.6661	0.8206	0.8100	0.7974
Recall	0.6602	0.7560	0.8065	0.7391	0.7676	0.7531	0.7831	0.7875	0.7372	0.7788	0.8000	0.8016

Fig. 7. Aggregated performance scores for all models (NSL-KDD dataset).

deep reinforcement learning techniques to perform the comparison, including: Logistic Regression, Support Vector Machine (SVM) with linear kernel and Radial Basis Function (RBF) kernel, Random Forest (RF), Gradient Boosting Machine (GBM), AdaBoost with simple trees as weak learners, MLP, Convolutional Neural Network (CNN), and several well-known DRL algorithms: DDQN [35,36], dueling [37] and A3C [38].

All the results presented in this paper are based on the test sets defined in Section 3.1. To analyze the prediction performance for the different models, and considering the highly unbalanced distribution of labels, we provide the following performance metrics: accuracy, F1 score, precision and recall. We base our definition of these performance metrics on the usually accepted ones [1]. Regarding highly unbalanced datasets, F1 is considered a better metric for prediction performance than accuracy, precision and recall. This metric (F1) will be used to rank the different algorithms applied to the two datasets.

When facing a multi-class classification problem, results can be presented in two ways: 'aggregated' and 'one vs. rest'. Considering the one vs. rest approach, we focus on a particular class (label) and consider the other classes as a single alternative class, simplifying the problem to a binary classification task for each particular class (one by one). In the case of aggregated results, we try to give a summary result for all classes. There are different alternatives to perform the aggregation (micro, macro, samples, weighted), varying in how the averaging process is done [39]. For the results presented in this paper, we have used the weighted average provided by scikit-learn [39], to calculate the aggregated F1, precision and recall scores.

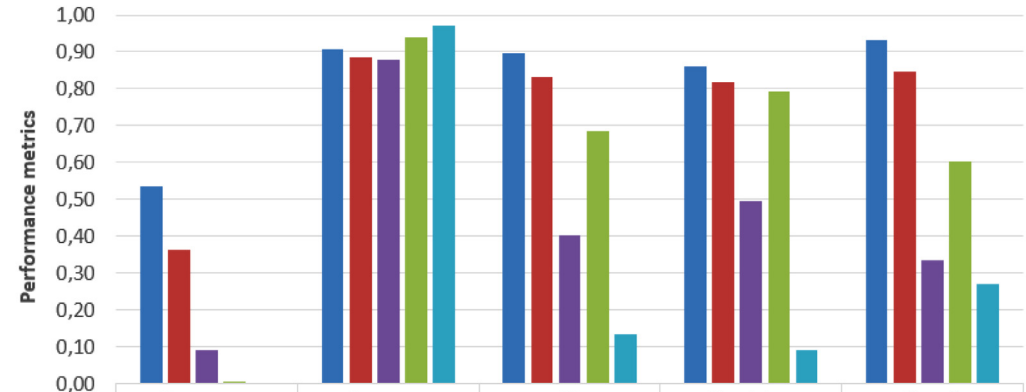
Aggregated performance metrics are summarized in Fig. 7 for the NSL-KDD dataset. In addition, prediction and training times are essential for IDS, since traffic is permanently changing. Good prediction metrics by themselves are not enough for deciding the best model. In Fig. 10 the computing times required for the training and prediction phase for all models are shown.

The best results for the F1 aggregated metric (for the NSL-KDD) are obtained by SVM-RBF, followed closely by AE-RL and the rest of the models notably behind (Fig. 7). Accuracy scores follow a similar pattern. We can observe that AE-RL provides an F1 score very similar to SVM-RBF (Fig. 7), but its main advantage comes from requiring much less time resources to perform prediction, as becomes evident from the prediction times for all models shown in Fig. 10.

Fig. 8 provides one vs. rest scores for the different labels, for the AE-RL model applied to the NSL-KDD dataset. We can observe how the AE-RL algorithm performs in a way that maximizes the detections of the less frequent labels. It can be seen that the accuracy for all labels is high, however the results are bounded by the false positives of these labels as shown by the F1 score. This model provides better results by slightly increasing the number of false positives while greatly reducing false negatives which is preferred in an IDS scenario. We can see in Fig. 8 the high values of the metrics F1 (> 0.83) and accuracy (> 0.88) for the labels that are not extremely unbalanced.

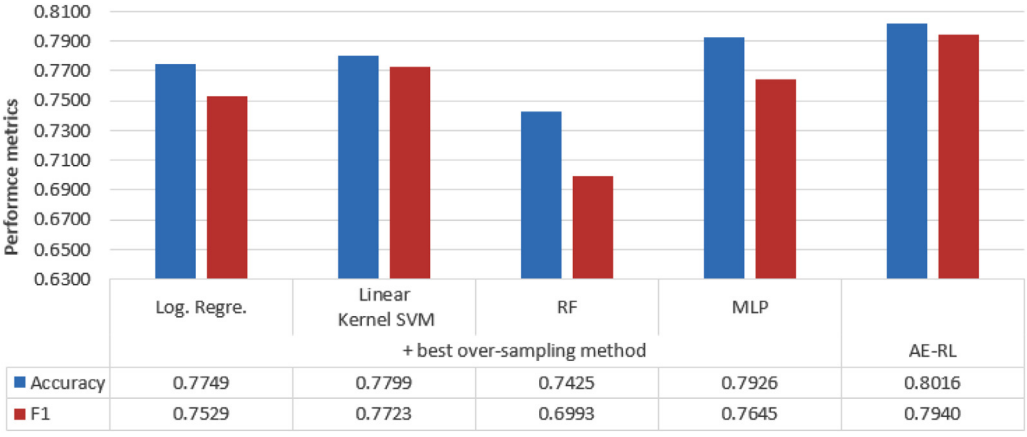
Fig. 9 presents the performance metrics of AE-RL (for the NSL-KDD dataset) compared to the performance metrics of other ML algorithms combined with an over-sampling method. The values for this figure were obtained from [34], which provides a thorough analysis of the improvements obtained by different ML algorithms (Logistic Regression, Linear SVM, RF and MLP) when an over-sampling method is applied to the less frequent categories of the dataset. The over-sampling methods considered in [34] are: VGM, SMOTE, SMOTE Borderline, SMOTE+ENN, SMOTE+Tomek, SMOTE SVM, Easy Ensemble and ADASYN. We have chosen the performance metrics obtained with the best possible over-sampling method when used in conjunction with each of the ML algorithms. The over-sampling methods have been used to increase the number of samples of the minority classes used for training. We can observe, in Fig. 9, that AE-RL outperforms the rest of ML algorithms even when they are used in combination with the best over-sampling method.

Fig. 10 provides the prediction and training times for all models for the NSL-KDD dataset. Considering prediction times, as expected, Linear-SVM and Logistic regression present the best prediction times, with models based on DRL and AE-RL with similar levels. Linear-SVM and Logistic Regression have been implemented using a simple shallow neural network with linear activation func-



	Frequency	Accuracy	F1	Precision	Recall
NORMAL	53,46%	0,9056	0,8948	0,8603	0,9322
DOS	36,46%	0,8873	0,8326	0,8183	0,8474
PROBE	9,25%	0,8775	0,4019	0,4976	0,3371
R2L	0,79%	0,9405	0,6851	0,7930	0,6031
U2R	0,04%	0,9697	0,1365	0,0914	0,2700

Fig. 8. One vs. Rest performance scores for the AE-RL model applied to the NSL-KDD dataset.



Accuracy	0.7749	0.7799	0.7425	0.7926	0.8016
F1	0.7529	0.7723	0.6993	0.7645	0.7940

Fig. 9. Performance metrics of AE-RL compared to the performance metrics of other ML algorithms + over-sampling method (NSL-KDD).

tions, hence their low prediction times. Likewise, all DRL models including AE-RL, are implemented with a simple non-linear neural network that also produces low prediction times. We can see that AE-RL has a prediction performance similar to SVM-RBF (Fig. 7), but in contrast it requires much shorter prediction times, and also shorter training times. An algorithm with short training and prediction times may be critical in deploying an IDS in the new highly demanding data networks. Moreover, short prediction times is essential for online prediction, hence the relevance of AE-RL in such real-time scenarios.

Fig. 11 provides an interesting chart that shows the evolution of the final F1 metric for the classifier, for different lower bounds of the environment agent's epsilon. The lower bound of epsilon is reached during training, starting from a high value (close to 1). As we can observe in Fig. 11, the best F1 value is obtained for a lower bound of the environment agent's epsilon of around 0.8, which means that we must keep a high exploration rate for the environment agent throughout the training in order to obtain better classification results.

Fig. 12 shows the confusion matrix of the AE-RL model for the AWID dataset, compared to two well-known algorithms: (a) an MLP with a structure similar to the one that implemented the policy network in AE-RL, and (b) the J48 algorithm with the results presented in [13], where J48 achieved the best classification results. The confusion matrix for AE-RL is shown on the left in Fig. 12 (chart A), while the middle and right positions are occupied by MLP and J48 (charts B and C, respectively). We can see that AE-RL has the least false negatives for the impersonation and flooding attacks. It is interesting to analyze the results of J48 that, regardless of its excellent aggregate results (Fig. 13), has 93% false negatives for the impersonation attack, which is an undesirable behavior for a detection system where a false negative is critical, since it implies an intrusion that has not been detected.

AE-RL tries to improve the classification of the less frequent classes. This can be visualized in Fig. 12, where AE-RL is able to reduce the false negatives for these less frequent classes, at the expense of a higher value of false positives for the normal class.

Fig. 13 presents the aggregated performance metrics of AE-RL for the AWID dataset. It also presents the aggregate results for the MLP, J48 and the rest of the models studied in [13]. The configuration of the models used for Fig. 12 remains the same as for Fig. 13.

We can see that considering the accuracy metric, J48 presents the best results. This is congruent with Fig. 12 since J48 presents zero false positives for the normal class, which is by far the most frequent class. However, we know this does not imply better

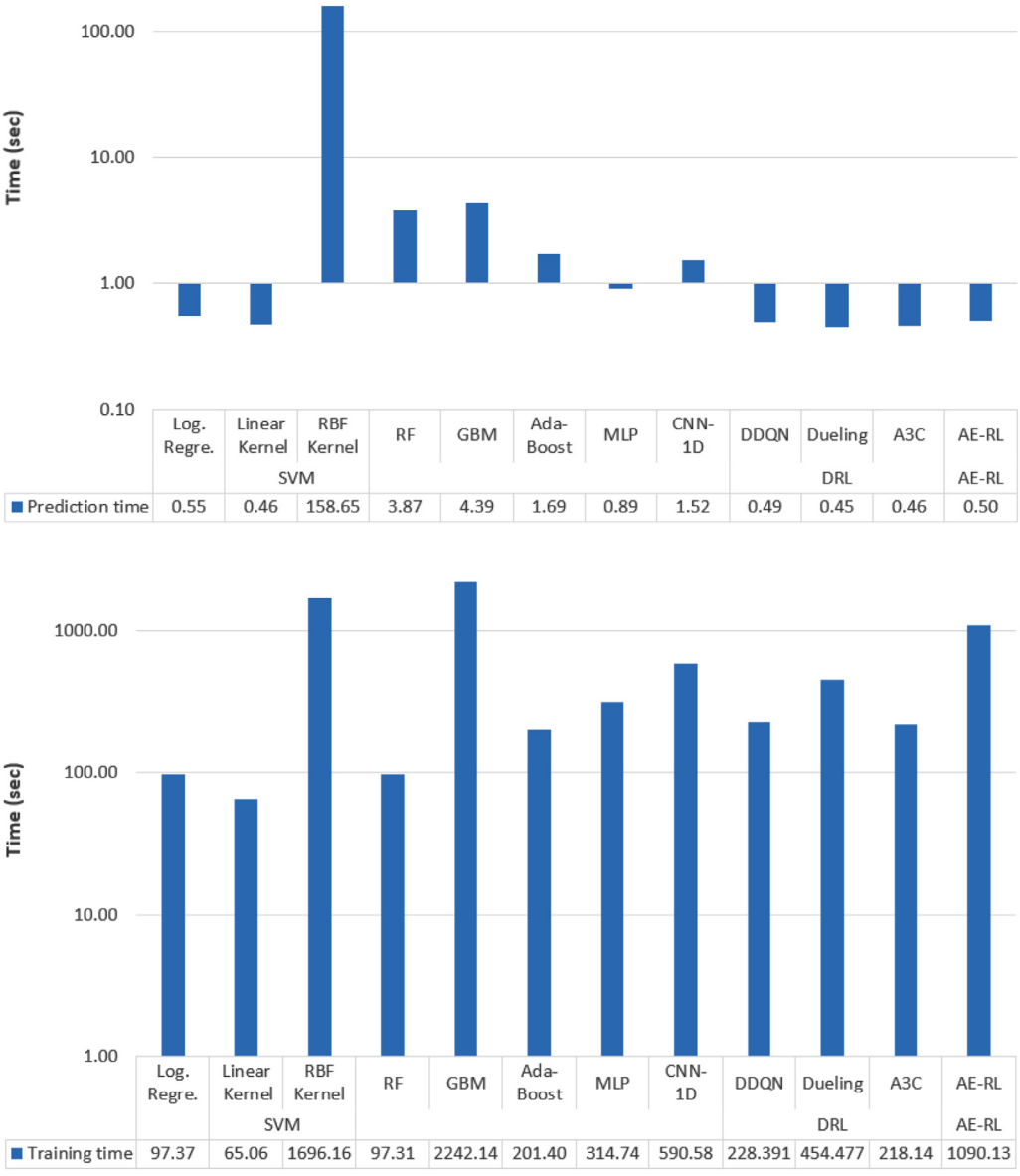


Fig. 10. Prediction times (upper chart) and training times (lower chart) for all learning models in logarithmic scale (NSL-KDD dataset).

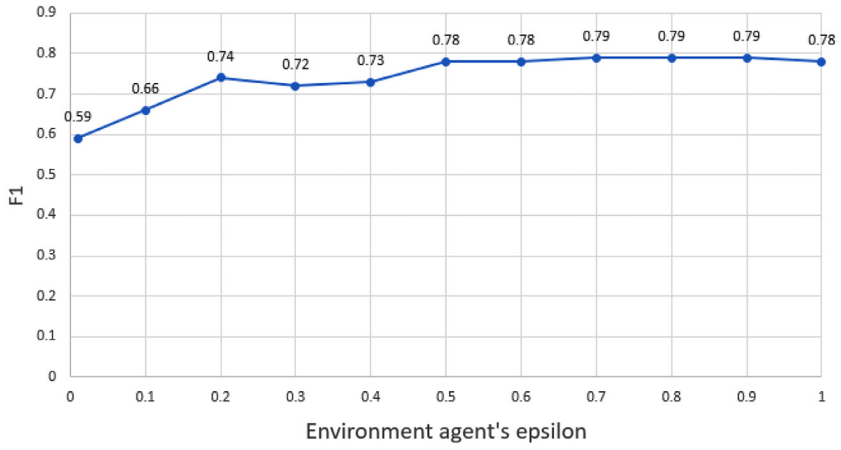


Fig. 11. Evolution of the final F1 metric for different lower bounds of the environment agent's epsilon (NSL-KDD).

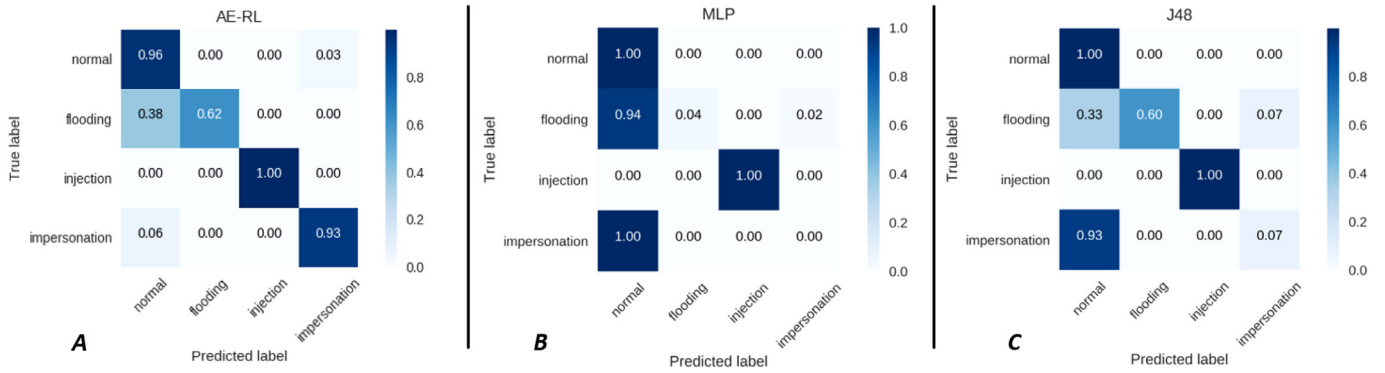
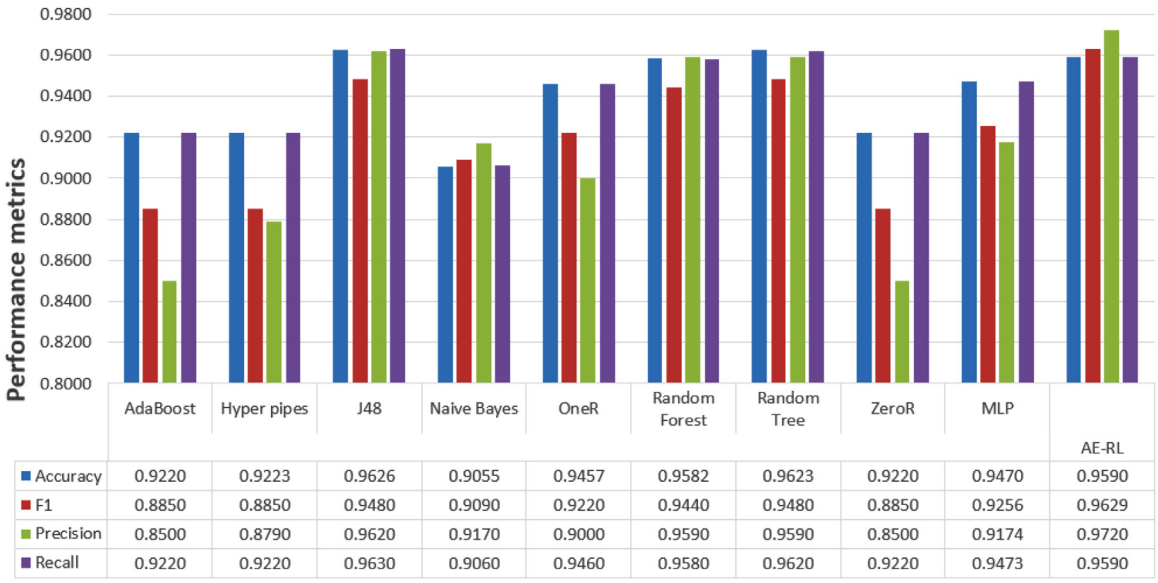


Fig. 12. Confusion matrix for the AE-RL algorithm compared to two other well-known algorithms (AWID dataset).



	AdaBoost	Hyper pipes	J48	Naive Bayes	OneR	Random Forest	Random Tree	ZeroR	MLP	AE-RL
Accuracy	0.9220	0.9223	0.9626	0.9055	0.9457	0.9582	0.9623	0.9220	0.9470	0.9590
F1	0.8850	0.8850	0.9480	0.9090	0.9220	0.9440	0.9480	0.8850	0.9256	0.9629
Precision	0.8500	0.8790	0.9620	0.9170	0.9000	0.9590	0.9590	0.8500	0.9174	0.9720
Recall	0.9220	0.9220	0.9630	0.9060	0.9460	0.9580	0.9620	0.9220	0.9473	0.9590

Fig. 13. Aggregated performance scores of AE-RL compared to other ML algorithms (AWID dataset).

performance: in fact, the F1 metric is greater for AE-RL since this metric is more adequate for unbalanced datasets, like the AWID.

The following points can be considered as a summary of the results obtained by AE-RL:

- AE-RL provides classification results similar to state-of-the-art classifiers.
- AE-RL requires less training time and much less prediction time than similar algorithms, which makes AE-RL ideally suitable for online prediction.
- AE-RL is especially suitable for unbalanced datasets, since it is specifically designed to avoid classification errors for under-represented classes.
- AE-RL is especially important in scenarios where false negatives are critical. False negatives are usual within the less frequent classes, since the algorithms try to obtain the best global prediction performance, which is more easily achieved by increasing the prediction of the most frequent classes, which implies more false negatives in the least frequent ones. This is actually the scenario for intrusion detection problems, where attacks correspond to the least frequent classes.

It is important to mention several details about the different models applied in the study. For SVM with linear kernel, we have used the primal solution which provides a much faster implementation in our specific case (high number of samples and small number of features). For the SVM with RBF, we used the dual implementation instead. We have implemented the MLP with several hidden layers (3 layers with 1024, 512 and 128 nodes). Considering the CNN model [40], we have applied the one-dimensional CNN due to the one-dimensional nature of the features.

We have implemented all the models in python using the scikit-learn package [39], except all linear models (including linear-SVM and Multinomial Logistic Regression), MLP, CNN and all DRL models for which we have used Tensorflow. To implement AE-RL, we have also used Tensorflow and Keras with custom code to sample the dataset. All computations have been performed in a PC with an Intel i7 CPU and 16GB RAM.

The code for the different models, as well as the hyperparameters values, can be found in the GitHub repository cited in [41].

5. Conclusion

IDS is a critical service for modern data networks. Network security attacks are complex, cannot be easily assigned to network patterns and are constantly changing. The special nature of network intrusion detection makes it necessary to investigate new models that can address some of the difficulties imposed by IDS on a machine learning algorithm, such as: noisy, unbalanced and complex datasets under changing conditions. This work tries to provide a new alternative, in the form of a model that brings the RL framework to the IDS problematic. The RL algorithms have

been particularly successful in other areas (e.g. robotic, finance, videogames, business operations…) and we are able to prove that they may also be available for IDS.

The proposed new model (AE-RL) integrates the reinforcement and supervised frameworks, providing a simulated environment that follows the guidelines of an RL environment. The resulting environment is able to: 1) interact with a dataset of pre-recorded samples formed by network features and associated intrusion labels, and 2) select samples with an optimized policy to achieve the best possible classification results. The specific learning mechanism provided to the environment is based on an adversarial strategy.

This work presents the first application of adversarial RL for intrusion detection and provides a novel technique that incorporates the environment's behavior into the learning process of a modified RL algorithm. It also proposes a new oversampling mechanism for the categories with worst performance, which has proven to be beneficial for training the RL algorithm.

As a summary, the contributions of this paper are: (1) Present a novel architecture based on a combination of supervised and adversarial RL models, making it a perfect alternative for prediction problems in highly demanding networks. (2) Demonstrate that the proposed new model has a prediction performance similar to highly non-linear models (e.g. SVM-RBF), but with the remarkable advantage of requiring much less computation time for prediction. (3) Propose a model that can be trained with a differentiable loss function in modern high-performance platforms (e.g. Tensorflow), even when the training is performed using a reward function that is not required to be differentiable, hence providing higher flexibility to the resulting classifier. (4) Provide a thorough comparison of our proposed model with several Machine Learning (ML) models for the problem of intrusion detection in networking (IDS), with a focus on prediction performance, prediction times and model flexibility. (5) Present a model that is particularly suitable for highly unbalanced datasets since the samples corresponding to incorrectly classified labels are more often presented for training.

As future lines of work, we plan the possible use of the intelligent environment in other classification tasks. Similarly, an interesting variant of the environment can be its implementation as a prioritized experience replay application [42], where sampling prioritization would be optimized using another agent, trained with the same reinforcement learning algorithms.

Conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2019.05.013.

References

[1] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools, IEEE Commun. Survey Tutor. 16 (1) (2014) 2014 First Quarter.

[2] M. Wang, et al., Machine learning for networking: workflow, advances and opportunities, IEEE Netw. 32 (2) (2018) 92–99.

[3] S. Kim et al., Building resilient and autonomous systems for iot network management - advantages and difficulties in adopting machine learning techniques, Internet Engineering Task Force (IETF), 6Lo Working Group, Internet-Draft: draft-kim-ml-iot-00, Seoul National University, 2018.

[4] K.A.P. da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of things: a survey on machine learning-based intrusion detection approaches, Comp. Netw. 151 (2019) 147–157. https://doi.org/10.1016/j.comnet.2019.01.023.

[5] M.M.E. Mahmoud, J.J.P.C. Rodrigues, S.H. Ahmed, S.C. Shah, J.F. Al-Muhtadi, V. Korotaev, V.H.C. de Albuquerque, Enabling technologies on cloud of things for smart healthcare, IEEE Access 6 (2018) 31950–31967, doi:10.1109/ACCESS.2018.2845399.

[6] J.J.P.C. Rodrigues, D.B De Rezende Segundo, H.A. Junqueira, M.H. Sabino, R.M. Prince, J. Al-Muhtadi, V.H.C. De Albuquerque, Enabling technologies for the internet of health things, IEEE Access 6 (2018) 13129–13141, doi:10.1109/ACCESS.2017.2789329.

[7] K. Arulkumaran, Deep reinforcement learning: a brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38.

[8] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 1st ed., A Bradford Book, March 1, 1998.

[9] N.V. Chawla, et al., SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.

[10] H. Bai, et al., ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: IEEE International joint conference on neural networks (IEEE world congress on computational intelligence), 2008, pp. 1322–1328.

[11] Z. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman & Hall/CRC, 2012 Book.

[12] M. Tavallaee et al., A detailed analysis of the KDD CUP 99 data set, Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009), pages 53–58.

[13] C. Kolias, et al., Intrusion detection in 802.11 networks, IEEE Commun. Surveys Tutor. 18 (1) (2016) first quarter.

[14] R.R. Guimaraes, L.A. Passos, R.H. Filho, V.H.C. de Albuquerque, J.J.P.C. Rodrigues, M.M. Komarov, J.P. Papa, Intelligent network security monitoring based on optimum-path forest clustering, IEEE Netw. (2018), doi:10.1109/MNET.2018.1800151.

[15] B. Ingre, A. Yadav, Performance analysis of NSL-KDD dataset using ANN, SPACES-2015, Dept of ECE, K L University, 2015.

[16] Y. Wahb, Improving the performance of multi-class intrusion detection systems using feature reduction, IJCSI Int. J. Comp. Sci. Issues 12 (3) (2015) May 201.

[17] M. Lopez-Martin, et al., Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT, Sensors 17 (9) (2017) 1967.

[18] A. Servin, Towards Traffic Anomaly Detection via Reinforcement Learning and Data Flow, Department of Computer Science, University of York, United Kingdom, 2007.

[19] X. Xu, Sequential anomaly detection based on temporal-difference learning: principles, models and case studies, Appl. Soft Comput. 10 (3) (2010) 859–867.

[20] A.V. Sukhanov, S.M. Kovalev, V. Stýskala, Advanced temporal-difference learning for intrusion detection, IFAC-PapersOnLine 48 (4) (2015) 43–48.

[21] X. Xu and T. Xie, A reinforcement learning approach for host-based intrusion detection using sequences of system calls. Advances in Intelligent Computing. ICIC 2005. Lecture Notes in Computer Science, vol 3644. Springer, Berlin.

[22] R.S. Sutton, Learning to predict by the methods of temporal differences, Mach. Learn. 3 (9) (1988) 44.

[23] A. Servin, Multi-Agent Reinforcement Learning for Intrusion Detection. Ph.D. thesis, University of York, 2009.

[24] K. Malialis, Distributed Reinforcement Learning for Network Intrusion Response. Ph.D. thesis, University of York, 2014.

[25] M.A. Wiering, et al., Reinforcement learning algorithms for solving classification problems, in: IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Paris, 2011, pp. 91–96.

[26] M.G. Lagoudakis, R. Parr, Reinforcement learning as classification: leveraging modern classifiers, in: Pro-ceedings of the 20th International Conference on MachineLearning (ICML-03), Washington, DC, USA, 2003, pp. 424–431.

[27] K.H. Quah, C. Quek, G. Leedham, Pattern classification using fuzzy adaptive learning control network and reinforcement learning, in: Proceedings of the 9th International Conference on Neural Information Processing. ICONIP '02, 3, Singapore, 2002, pp. 1439–1443. 2002.

[28] L. Pinto et al., Robust adversarial reinforcement learning, arXiv:1703.02702v1 [cs.LG], 2017.

[29] P. Hernandez-Leal et al., A survey of learning in multiagent environments: dealing with non-stationarity, arXiv:1707.09183v1 [cs.MA], 2017.

[30] L. Busoniu, R. Babuska, B. De Schutter, Multi-agent reinforcement learning: an overview, in: D. Srinivasan, L.C. Jain (Eds.), Chapter 7 in Innovations in Multi-Agent Systems and Applications – 1, Springer, Berlin, Germany, 2010, pp. 183–221. vol. 310 of Studies in Computational Intelligence.

[31] S. Huang et al., Adversarial attacks on neural network policies, arXiv:1702.02284 [cs.LG], 2017.

[32] R. Elderman et al., "Adversarial reinforcement learning in a cyber security simulation" International Conference on Agents and Artificial Intelligence (ICAART 2017).

[33] M. Zhu, Z. Hu, P. Liu, Reinforcement learning algorithms for adaptive cyber defense against heartbleed, in: Proceedings of the First ACM Workshop on Moving Target Defense, 2014, pp. 51–58.

[34] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, Variational data generative model for intrusion detection, Knowl. Inform. Syst. (2018). https://doi.org/10.1007/s10115-018-1306-7.

[35] V. Mnih et al. Playing atari with deep reinforcement learning, arXiv:1312.5602 [cs.LG], 2013.

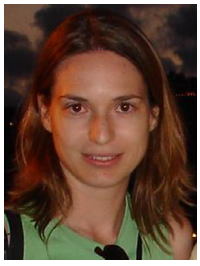
[36] H. Van Hasselt et al., Deep reinforcement learning with double Q-learning, arXiv:1509.06461 [cs.LG], 2015.

[37] Z. Wang et al., Dueling network architectures for deep reinforcement learning, arXiv:1511.06581v3 [cs.LG], 2016.

[38] I. Grondman, A survey of actor-critic reinforcement learning: standard and natural policy gradients, IEEE Trans. Syst. Man Cybernet. Part C (Appl. Rev.) 42 (6) (2012) 1291–1307.
[39] F. Pedregosa, et al., Scikit-learn: machine learning in python, JMLR 12 (2011) 2825–2830.
[40] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016 Book. Ch 9.
[41] G. Caminero, Anomaly-ReactionRL, GitHub, 2019. https://github.com/gcamfer/Anomaly-ReactionRL.
[42] T. Schaul et al. Prioritized experience replay, arXiv:1511.05952v4 [cs.LG], 2016.

Guillermo Caminero received its M.Sc. in Telecommunications Engineering at Universidad de Valladolid, Spain in 2018. He is a collaborator of the research team of the Communications Systems and Networks (SRC) Laboratory at Universidad de Valladolid, working on the application of machine learning to intrusion detection in data networks.

Manuel Lopez-Martin is a research associate and Ph.D. candidate at Universidad de Valladolid (Spain). His research activities involve applying machine learning to intrusion detection in data networks and time-series forecasting. He received his M.Sc. in Telecommunications Engineering in 1985 from UPM-Madrid and his M.Sc. in Computer Sciences in 2013 from UAM-Madrid. He has worked as a data scientist at Telefonica and has more than 25 years of experience in the development of IT software projects



Belen Carro received a Ph.D. degree in the field of broadband access networks from the Universidad de Valladolid, Spain, in 2001. She is Professor and Director of the Communications Systems and Networks (SRC) laboratory at Universidad de Valladolid, working as Research Manager in NGN communications and services, VoIP/QoS and machine learning. She has supervised a dozen Ph.D. students and has extensive research publications experience, as author, reviewer and editor.